



TITLE:

# A Note on Alternating Pushdown Automata With Sublogarithmic Space

AUTHOR(S):

Xu, Jianliang; Inoue, Katsushi; Wang, Yue; Ito, Akira

---

CITATION:

Xu, Jianliang ...[et al]. A Note on Alternating Pushdown Automata With Sublogarithmic Space. 数理解析研究所講究録 1995, 906: 55-63

ISSUE DATE:

1995-04

URL:

<http://hdl.handle.net/2433/59461>

RIGHT:

## A Note on Alternating Pushdown Automata With Sublogarithmic Space

Jianliang Xu   Katsushi Inoue  
Yue Wang   Akira Ito

Department of Computer Science and Systems Engineering, Faculty of Engineering  
Yamaguchi University, Ube, 755 Japan

### Abstract

This paper investigates some fundamental properties of alternating one-way (or two-way) pushdown automata (pda's) with sublogarithmic space.

Let  $strong-2APDA(L(n))$  ( $strong-2DPDA(L(n))$ ,  $strong-2NPDA(L(n))$ ,  $strong-2UPDA(L(n))$ ) denote the class of languages accepted by strongly  $L(n)$  space-bounded two-way alternating pda's, (deterministic pda's, non-deterministic pda's, alternating pda's with only universal states), and let  $weak-2DPDA(L(n))$  ( $weak-2NPDA(L(n))$ ,  $weak-2UPDA(L(n))$ ) denote the class of languages accepted by weakly  $L(n)$  space-bounded two-way deterministic pda's (nondeterministic pda's, alternating pda's with only universal states), and let  $weak-1APDA(L(n))$  ( $weak-1SPACE(L(n))$ ) denote the class of languages accepted by weakly  $L(n)$  space-bounded one-way alternating pda's (alternating Turing machines).

We first show that  $strong-2APDA(\log \log n) - weak-1SPACE(o(\log n)) \neq \emptyset$ , and  $weak-1APDA(\log \log n) - (weak-2NPDA(o(\log n)) \cup weak-2UPDA(o(\log n))) \neq \emptyset$ . Then, we show that for any function  $\log \log n \leq L(n) = o(\log n)$ ,  $weak-1APDA(L(n))$  and  $X-YPDA(L(n))$  ( $X \in \{strong, weak\}$  and  $Y \in \{2D, 2N, 2U\}$ ) are not closed under concatenation, Kleene closure, and length preserving homomorphism.

Key words: Alternating Pushdown Automata, Sublogarithmic space complexity, One-way versus two-way

## 1 Introduction

Recently, many investigations about alternating Turing machines with sublogarithmic space have been made [2,4,9,10,13,15]. It is shown in [9] that for any function  $\log \log n \leq L(n) = o(\log n)$ ,  $L(n)$  space-bounded two-way alternating Turing machines are more powerful than  $L(n)$  space-bounded one-way alternating Turing machines. Iwama [10] showed that  $o(\log \log n)$  space-bounded two-way alternating Turing machines accept only regular languages. Chang, Ibarra and Ravikumar [4] showed that there is a language over a unary alphabet that can be accepted by a weakly  $\log \log n$  space-bounded one-way alternating Turing machine, but not by any two-way nondeterministic Turing machine with  $o(\log n)$  space. Szeietowski [15] showed that there is a language accepted by a weakly  $\log \log n$  space-bounded one-way alternating Turing machine, but not by any strongly  $o(\log n)$  space-bounded two-way alternating Turing machine. Braunmühl, Gengler and Rettinger [2], and Liśkiewicz and Reischuk [13] showed that the alternation hierarchy for Turing machines with space bounds between  $\log \log n$  and  $\log n$  is infinite. (Note the fact that all alternation hierarchies related to space-bounded two-way Turing machines collapse, provided we consider strong space-complexity and space-bounds in  $\Omega(\log n)$ . This is because the class of languages accepted by strongly  $L(n)$  space-bounded two-way nondeterministic Turing machines is closed under complementation for  $L(n) = \Omega(\log n)$  [8,14].) There have been few investigations about pushdown automata with small space, especially with sublogarithmic space. Gabarro [7] showed that (i) there are languages with pushdown complexity strictly in  $n^{1/q}$  or  $\log n$  ( $q \geq 2$ ), and (ii) the family of languages accepted by one-way nondeterministic pushdown automata with sublinear space is a full-A.F.L. containing one infinite decreasing chain of full-A.F.L.'s. Duris and Galil [5] showed that (i) for any function  $\log \log n \leq L(n) = o(n)$ ,  $L(n)$  space-bounded two-way deterministic pushdown automata are less powerful than  $L(n)$  space-bounded two-way deterministic Turing machines, (ii)  $o(n)$  space-bounded two-way deterministic pushdown automata accept only regular languages over a unary alphabet, and (iii) there is a non-regular language accepted by a strongly  $\log \log n$  space bounded two-way deterministic pushdown automaton. Yoshinaga and Inoue [16] investigated several properties of alternating multi-counter automata with sublinear space.

This paper investigates some fundamental properties of alternating one-way (or two-way) pushdown automata with sublogarithmic space.

Section 2 gives the definitions and notations necessary for this paper. Let  $strong-2APDA(L(n))$  ( $strong-2DPDA(L(n))$ ,  $strong-2NPDA(L(n))$ ,  $strong-2UPDA(L(n))$ ) denote the class of languages accepted by strongly  $L(n)$  space-bounded two-way alternating pushdown automata (deterministic pushdown automata, nondeterministic pushdown automata, alternating pushdown automata with only universal states), and let  $weak-2DPDA(L(n))$  ( $weak-2NPDA(L(n))$ ,  $weak-2UPDA(L(n))$ ) denote the class of languages accepted by weakly  $L(n)$  space-bounded two-way deterministic pushdown automata (nondeterministic pushdown automata, alternating pushdown automata with only universal states). Furthermore, let  $weak-1APDA(L(n))$  ( $weak-1SPACE(L(n))$ ,  $weak-1UPSPACE(L(n))$ ) denote the class of languages accepted by weakly  $L(n)$  space-bounded one-way alternating pushdown automata (alternating Turing machines, alternating Turing machines with only universal states).

Section 3 investigates a relationship between the accepting powers of one-way and two-way alternating pushdown automata with sublogarithmic space, and shows that  $strong-2APDA(\log \log n) - weak-1SPACE(o(\log n)) \neq \emptyset$  (and thus  $strong-2APDA(\log \log n) - weak-1APDA(o(\log n)) \neq \emptyset$ ). This result strengthens the fact [9] that  $strong-SPACE(\log \log n) - weak-1SPACE(o(\log n)) \neq \emptyset$ .

Section 4 investigates a relationship among the accepting powers of alternating pushdown automata, nondeterministic pushdown automata and alternating pushdown automata with only universal states with sublogarithmic space, and shows, for example, that  $weak-1APDA(\log \log n) - (weak-NSPACE(o(\log n)) \cup weak-USPACE(o(\log n))) \neq \emptyset$ , and thus  $weak-1APDA(\log \log n) - (weak-2NPDA(o(\log n)) \cup weak-2UPDA(o(\log n))) \neq \emptyset$ . This result strengthens the fact [9] that

$weak-1SPACE(\log \log n) - (weak-NSPACE(o(\log n)) \cup weak-USPACE(o(\log n))) \neq \emptyset$ . We also show that for any function  $\log \log n \leq L(n) = o(\log n)$ ,  $weak-1NSPACE(L(n))$  and  $weak-1USPACE(L(n))$  is incomparable. This result solves an open problem in [9].

Section 5 investigates several fundamental closure properties, and shows that, for any function  $\log \log n \leq L(n) = o(\log n)$ ,  $weak-1APDA(L(n))$  and  $X-YPDA(L(n))$  ( $X \in \{strong, weak\}$  and  $Y \in \{2D, 2N, 2U\}$ ) are not closed under concatenation, Kleene closure, and length preserving homomorphism.

Section 6 briefly states a relationship between 'strong' and 'weak'.

## 2 Preliminaries

We assume that the reader is familiar with the basic concepts and terminology concerning alternating machines and computational complexity. (If necessary, see [3,9,13].)

A *two-way alternating pushdown automaton* (2APDA) is a generalization of a two-way nondeterministic pushdown automaton (2NPDA) [11] whose state set is partitioned into 'universal' and 'existential' states. The input of a 2APDA  $M$  is delimited by the left endmarker  $\#$  and the right endmarker  $\$$ . We can view the computation of  $M$  as a tree whose nodes are labelled by instantaneous descriptions (ID's). An ID is called *universal* (*existential*, *accepting*) if the state associated with that ID is universal (existential, accepting). A *computation tree* of  $M$  on input  $x$  is a tree, such that the root is labelled by the initial ID and the children of any nonleaf node labelled by a universal (existential) ID include all (one) of the immediate successors of that ID. A computation tree is *accepting* if it is finite and all the leaves are labelled by accepting ID's.  $M$  accepts  $x$  if there is an accepting tree of  $M$  on  $x$ . A computation tree of  $M$  (on some input) is  $l$  space-bounded if all nodes of the tree are labelled with ID's using at most  $l$  cells of the pushdown stack. Let  $L(n)$  be a function.  $M$  is *weakly*  $L(n)$  space-bounded if for every input  $x$  of length  $n$ ,  $n \geq 1$ , that is accepted by  $M$ , there exists an  $L(n)$  space-bounded accepting computation tree of  $M$  on  $x$ .  $M$  is *strongly*  $L(n)$  space-bounded if for every input  $x$  of length  $n$  (accepted by  $M$  or not),  $n \geq 1$ , any computation tree of  $M$  on  $x$  is  $L(n)$  space-bounded.

A *one-way alternating pushdown automaton* (1APDA) is a 2APDA whose input head cannot move to the left. We denote by 2UPDA (1UPDA) a 2APDA (1APDA) whose states are all universal. A one-way nondeterministic pushdown automaton (1NPDA) is a 1APDA whose states are all existential. Of course, a 2NPDA is a 2APDA whose states are all existential. A two-way (one-way) deterministic pushdown automaton, denoted by 2DPDA (1DPDA), is a 2APDA (1APDA) whose ID's each have at most one successor.

For each  $X \in \{2A, 1A, 2U, 1U, 2N, 1N, 2D, 1D\}$ , let *strong-XPDA*( $L(n)$ ) denote the class of sets accepted by strongly  $L(n)$  space-bounded XPDA's, and *weak-XPDA*( $L(n)$ ) denote the class of sets accepted by weakly  $L(n)$  space-bounded XPDA's.

A two-way (one-way) alternating Turing machine, denoted by 2ATM (1ATM), has a two-way (one-way) read-only input tape (with the left endmarker  $\#$  and the right endmarker  $\$$ ) and a separate two-way read-write storage-tape.

Let  $L(n)$  be a function and  $M$  be a 2ATM. The concepts of 'a computation tree of  $M$ ', ' $L(n)$  space-bounded accepting computation tree', and 'weakly (strongly)  $L(n)$  space-bounded' are defined as above.

We denote by 2UTM (1UTM) a 2ATM (1ATM) whose states are all universal. A two-way (one-way) nondeterministic Turing machine, denoted by 2NTM (1NTM), is a 2ATM (1ATM) whose states are all existential, and a two-way (one-way) deterministic Turing machine, denoted by 2DTM (1DTM), is a 2ATM (1ATM) whose ID's each have at most one successor. For each  $X \in \{A, U, N, D\}$ , let *strong-XSPACE*( $L(n)$ ) (*weak-XSPACE*( $L(n)$ )) denote the class of sets accepted by strongly (weakly)  $L(n)$  space-bounded 2XTM's, and *strong-1XSPACE*( $L(n)$ ) (*weak-1XSPACE*( $L(n)$ )) denote the class of sets accepted by strongly (weakly)  $L(n)$  space-bounded 1XTM's. This paper is mainly concerned with strongly and weakly  $o(\log n)$  space-bounded APDA's (ATM's).

Let  $M$  be a 2ATM, and  $S_M = Q \times (\Gamma - \{B\})^* \times N$ , where  $Q$  is the set of states of  $M$ ,  $\Gamma$  is the storage-tape alphabet of  $M$ ,  $B$  is the blank symbol, and  $N$  denote the set of all positive integers. An element  $(q, \alpha, j)$  of  $S_M$  is called a *storage state* of  $M$ , and represents the state of the finite control, the non-blank contents of the storage-tape, and the storage-head position.

We conclude this section by giving several notations used below.

**Notation 1.** For any string  $w$ ,  $|w|$  denotes the length of  $w$ , and  $w^R$  denotes the reversal (i.e., mirror image) of  $w$ . For any set  $S$ ,  $|S|$  denotes the number of elements of  $S$ .

**Notation 2.** For each integer  $n \geq 1$ , and for each integer  $i (1 \leq i \leq 2^n)$ , let  $B(n, i)$  denote the binary number of  $n$  bits with the leftmost bit as the most significant bit which represents the integer  $i - 1$ . Thus,  $B(3, 1) = 000$ ,  $B(3, 2) = 001$ ,  $B(3, 3) = 010 \dots$ ,  $B(2, 2^3) = B(3, 8) = 111$ .

**Notation 3.** For each integer  $n \geq 1$ , and for each integer  $i (1 \leq i \leq 2^{2^n})$ , let

$$W(n, i) \triangleq x_{i1} B(n, 1) x_{i2} B(n, 2) \dots x_{i2^n} B(n, 2^n), \text{ and}$$

$$W'(n, i) \triangleq \begin{cases} x_{i1} B(n, 1) x_{i2} B(n, 2) \dots x_{i2^n} B(n, 2^n), & \text{if } i \text{ is odd,} \\ x_{i1} B(n, 1)^R x_{i2} B(n, 2)^R \dots x_{i2^n} B(n, 2^n)^R, & \text{if } i \text{ is even,} \end{cases}$$

where  $x_{ij}'s \in \{a, b\}$ , and  $h(W(n, i)) = h(W'(n, i)) = B(2^n, i)$  (where  $h: \{0, 1, a, b\} \rightarrow \{0, 1\}$  is a homomorphism such that  $h(0) = h(1) = \lambda$ ,  $h(a) = 0$  and  $h(b) = 1$ ).

**Notation 4.**

For each integer  $n \geq 1$ , let:

$$Ruler_1(n) \triangleq W(n, 1) \# W(n, 2) \# \dots \# W(n, 2^{2^n}), \text{ and } Ruler_2(n) \triangleq W'(n, 1) \# W'(n, 2) \# \dots \# W'(n, 2^{2^n}).$$

Throughout this paper, let  $h$  denote the homomorphism described above, and let:

$$\begin{aligned} D(n) &= \{x_1 B(n, 1) x_2 B(n, 2) \cdots x_{2^n} B(n, 2^n) \mid \forall i (1 \leq i \leq 2^n) [x_i \in \{a, b\}]\} \text{ for each } n \geq 1, \text{ and} \\ D'(n) &= \{x_1 B(n, 1)^R x_2 B(n, 2)^R \cdots x_{2^n} B(n, 2^n)^R \mid \forall i (1 \leq i \leq 2^n) [x_i \in \{a, b\}]\} \text{ for each } n \geq 1. \end{aligned}$$

### 3 Two-way versus One-way

This section investigates a relationship between the accepting powers of one-way and two-way alternating pushdown automata with sublogarithmic space.

We first give some definitions necessary for proving Theorem 1 below. Let  $M$  be a 1ATM, and  $\Sigma$  be the input alphabet of  $M$ . For each storage state  $(q, \alpha, j)$  of  $M$  and for each  $w \in \Sigma^+$ , let a  $(q, \alpha, j)$ -computation tree of  $M$  on  $w$  be a computation tree which represents a computation of  $M$  on  $w$  starting with the input head on the leftmost position of  $w$  and with the storage state  $(q, \alpha, j)$ . A  $(q, \alpha, j)$ -accepting computation tree of  $M$  on  $w$  is a  $(q, \alpha, j)$ -computation tree whose leaves are all labelled with accepting ID's.

**Theorem 1.**

*strong-2APDA*( $\log \log n$ ) - *weak-1ASPACE*( $o(\log n)$ )  $\neq \emptyset$ .

**Proof.**

Let  $L_1 = \{Ruler_1(n) cucu_1 cu_2 c \cdots cu_k \in \{0, 1, a, b, c, \#\}^+ \mid n \geq 1 \ \& \ k \geq 1 \ \& \ u \in D'(n) \ \& \ \forall i (1 \leq i \leq k) [u_i \in D(n)] \ \& \ \exists r (1 \leq r \leq k) [h(u) = h(u_r)]\}$ .

To prove the theorem, we show that

(1)  $L_1 \in \text{strong-2APDA}(\log \log n)$ , and

(2)  $L_1 \notin \text{weak-1ASPACE}(o(\log n))$ .

(1): The set  $L_1$  will be accepted by a strongly  $\log \log n$  space-bounded 2APDA  $M$  which acts as follows.

We assume without loss of generality that an input string to  $M$  is of the form

$$Ruler_1(n) cucu_1 cu_2 c \cdots cu_k \quad \dots \dots (1)$$

for some  $n \geq 1$ , where  $k \geq 1$  and

- (i)  $u = y_1 v_1 y_2 v_2 \cdots y_l v_l$  (where  $l \geq 2$ ,  $y_j's \in \{a, b\}$ , and  $v_j's \in \{0, 1\}^+$ ), and
- (ii) for each  $i (1 \leq i \leq k)$ ,  $u_i = y_{i1} v_{i1} y_{i2} v_{i2} \cdots y_{il_i} v_{il_i}$  (where  $l_i \geq 2$ ,  $y_{ij}'s \in \{a, b\}$ , and  $v_{ij}'s \in \{0, 1\}^+$ ).

This is because it is shown in [5] that the set  $\{Ruler_1(n) \mid n \geq 1\}$  can be recognized by a strongly  $\log \log n$  space-bounded 2DPDA, and thus input strings of the form different from the above can easily be rejected by  $M$ .

After recognizing  $Ruler_1(n)$ ,  $M$  checks whether  $\forall j (1 \leq j \leq l) [v_j = B(n, j)^R]$ ,  $v_l = \underbrace{11 \cdots 1}_n$ , and  $\forall i (1 \leq i \leq k) [\forall j (1 \leq j \leq l_i) [v_{ij} = B(n, j)] \text{ and } v_{il_i} = \underbrace{11 \cdots 1}_n]$ . This check is deterministically done by using  $Z^n$  stored in the pushdown stack while  $M$  recognizes  $Ruler_1(n)$ , where  $Z$  is a pushdown stack symbol.

After this check,  $M$  existentially chooses some  $r (1 \leq r \leq k)$ , moves to the segment  $u_r$ , and universally checks whether  $y_{rj} = y_j$  for each  $1 \leq j \leq l_r$ . In order to check that  $y_{rj} = y_j$ ,  $M$  simply stores the symbol  $y_{rj}$  in the finite control, stores the "yardstick" string  $v_{rj} = B(n, j)$  (positioned just after  $y_{rj}$ ) in the pushdown stack, picks up the symbol  $y_j$  (this is deterministically done by using  $v_{rj}$  in the pushdown stack and the yardstick string  $v_j = B(n, j)^R$ ), and enters an accepting state only if it finds out that  $y_{rj} = y_j$ .

It will be obvious that each computation path of any computation tree of  $M$  on the input  $x$  of the form (1) is such that the space of the pushdown stack is bounded by  $n \leq \log \log |x|$ . (Note that  $M$  marks off  $n \leq \log \log |x|$  stack cells after recognizing  $Ruler_1(n)$ .)

(2): Suppose that there exists a weakly  $L(n)$  space-bounded 1ATM  $M$  accepting  $L_1$ , where  $L(n) = o(\log n)$ . Let  $s$  and  $k$  be the numbers of states (of the finite control) and storage-tape symbols of  $M$ , respectively. For each  $n \geq 1$ , let:

$$V(n) = \{Ruler_1(n) cucu_1 cu_2 c \cdots cu_{2^n} \in L_1 \mid \forall i (1 \leq i \leq 2^{2^n}) [u_i \in D(n)] \ \& \ u \in D'(n)\},$$

$$W(n) = \{cu_1 cu_2 c \cdots cu_{2^n} \mid \forall i (1 \leq i \leq 2^{2^n}) [u_i \in D(n)]\}.$$

For each  $x$  in  $V(n)$ , We have:

$$\begin{aligned} \text{(i)} \quad |x| &= |Ruler_1(n)| + |u| + (2^{2^n} + 1) + 2^{2^n} |u_i| = 2^{2^n} \cdot (n+1) \cdot 2^n + 2^n(n+1) + 2^{2^n} + 1 + 2^{2^n} \cdot (n+1) \cdot 2^n \\ &\triangleq r(n) = O(n \cdot 2^n \cdot 2^{2^n}) \end{aligned}$$

(ii) There exists an  $L(r(n))$  space-bounded accepting computation tree of  $M$  on  $x$ .

For each storage state  $(q, \alpha, j)$  of  $M$  and for each  $y$  in  $W(n)$ , let

$$M_y(q, \alpha, j) = \begin{cases} 1 & \text{if there exists an } L(r(n)) \text{ space-bounded} \\ & (q, \alpha, j)\text{-accepting computation tree of } M \\ & \text{on } y. \\ 0 & \text{otherwise.} \end{cases}$$

For any two strings  $y, z$  in  $W(n)$ , we say that  $y$  and  $z$  are *M-equivalent* if for each storage state  $(q, \alpha, j)$  of  $M$  with  $|\alpha| \leq L(r(n))$  and  $1 \leq j \leq |\alpha|$ ,  $M_y(q, \alpha, j) = M_z(q, \alpha, j)$ . Clearly, M-equivalence is an equivalence relation on strings in  $W(n)$ , and there are at most

$$E(n) = 2^{s \cdot [L(r(n))] \cdot k^{L(r(n))}}$$

M-equivalence classes denoted by  $C_1, C_2, \dots, C_{E(n)}$ .

For each  $y = cu_1cu_2c \dots cu_{2^n}$  in  $W(n)$ , let

$$b(y) = \{u \in D'(n) \mid \exists j(1 \leq j \leq 2^n)[h(u_j) = h(u)]\}.$$

Furthermore, for each  $n \geq 1$ , let  $R(n) = \{b(y) \mid y \in W(n)\}$ . Then  $|R(n)| = 2^{2^n} - 1$ .

Since  $\lim_{n \rightarrow \infty} L(n)/\log n = 0$ , it follows that  $\lim_{n \rightarrow \infty} L(r(n))/\log r(n) = 0$ . .... (2)

Since  $r(n) = O(n \cdot 2^n \cdot 2^{2^n})$ , it follows that for some constant  $a > 0$ ,  $\log r(n) < a \cdot 2^n$ . From this and equation (2), we have  $\lim_{n \rightarrow \infty} L(r(n))/a \cdot 2^n = 0$ . From this, it follows that  $\lim_{n \rightarrow \infty} L(r(n))/2^n = 0$ . So we have  $|R(n)| > E(n)$  for large  $n$ . For such  $n$ , there must be some  $Q, Q' (Q \neq Q')$  in  $R(n)$  and some  $C_i (1 \leq i \leq E(n))$  such that the following statement holds:

"There exists two strings  $y, z \in W(n)$  such that (i)  $b(y) = Q \neq Q' = b(z)$ , and (ii)  $y, z \in C_i$  (i.e.  $y$  and  $z$  are M-equivalent.)"

Because of (i), we can without loss of generality assume that there is some  $u$  such that  $u \in b(y) - b(z)$ . It is clear that  $y' = \text{Ruler}_1(n)cuy$  is in  $V(n)$ , so there exists an  $L(r(n))$  space-bounded accepting computation tree of  $M$  on  $y'$ . Because of (ii), From this tree, we can easily construct an  $L(r(n))$  space-bounded accepting computation tree of  $M$  on  $z' = \text{Ruler}_1(n)cuz$ . Thus, we can conclude that  $z'$  is also accepted by  $M$ . Since  $z'$  is not in  $L_1$ , We get a contradiction. This completes the proof of (2). Q.E.D.

### Theorem 2.

*strong-2DPDA*( $\log \log n$ ) - *weak-1NSPACE*( $o(\log n)$ )  $\neq \emptyset$

**Proof.**

Let  $L_2 = \{\text{Ruler}_1(n) \mid n \geq 1\}$ . Since  $L_2 \in \text{strong-2DPDA}(\log \log n)$  [5], to prove this theorem, it is sufficient to show that  $L_2$  is not in *weak-1NSPACE*( $o(\log n)$ ).

It is shown in [4] that for any  $L \in \text{weak-1NSPACE}(o(\log n))$ ,  $L$  satisfies the pumping property that, for large enough  $n$ , if  $w$  is such that  $|w| \geq n$  and  $w \in L$ , then there exist  $x, y$ , and  $z$  such that (1)  $w = xyz$ , and (2)  $xy^iz \in L$  for all  $i \geq 0$ . From this and the obvious fact that  $L_2$  does not satisfy the pumping property, it follows that  $L_2 \notin \text{weak-1NSPACE}(o(\log n))$ . This completes the proof. Q.E.D.

### Theorem 3.

*strong-2DPDA*( $\log \log n$ ) - *weak-1USPACE*( $o(\log n)$ )  $\neq \emptyset$

**Proof.** Let  $L_3 = \{\text{Ruler}_1(n)cucu' \mid n \geq 1 \text{ \& } u \in D(n) \text{ \& } u' \in D'(n) \text{ \& } h(u) \neq h(u')\}$ . We can show that  $L_3 \in \text{strong-2DPDA}(\log \log n)$ . (The proof is left to the reader.) We below prove that  $L_3$  is not in *weak-1USPACE*( $o(\log n)$ ).

Suppose that there exists a weakly  $L(n)$  space-bounded 1UTM,  $M$ , which accepts  $L_3$ , where  $L(n) = o(\log n)$ . For each  $u \in D(n)$ ,  $n \geq 1$ , there exists exactly one  $u' \in D'(n)$  such that  $h(u) = h(u')$ . We denote this  $u'$  by  $[u]$ . For each  $n \geq 1$ , let

$$V(n) = \{\text{Ruler}_1(n)cuc[u] \mid u \in D(n)\}.$$

For each  $x = \text{Ruler}_1(n)cuc[u]$  in  $V(n)$ , there is at least one computation path of  $M$  on  $x$  in which  $M$  never enters an accepting state, because  $x \notin L_3$ . Fix such a computation path of  $M$  on  $x$ , and denote it by  $p(x)$ . Let  $s(x)$  be the storage state of  $M$  just after the point where in  $p(x)$  the input head left the second 'c' of  $x$ . Then the following proposition must hold.

**Proposition 1.** For any two different strings  $x, y$  in  $V(n)$ ,  $s(x) \neq s(y)$ .

[ **Proof.** For otherwise, suppose that  $x = \text{Ruler}_1(n)cuc[u]$ ,  $y = \text{Ruler}_1(n)cuc[v]$ ,  $u \neq v$ , and  $s(x) = s(y)$ . Then, there would be a computation path of  $M$  on the string  $\text{Ruler}_1(n)cuc[v]$  in which  $M$  never enters an accepting state. This means that  $\text{Ruler}_1(n)cuc[v]$  is rejected by  $M$ . This contradicts the fact that  $\text{Ruler}_1(n)cuc[v]$  is in  $L_3$ . ]

**Proof of theorem 3 (continued).**

For each  $n \geq 1$ , let  $V'(n) = \{\text{Ruler}_1(n)cucu' \mid u \in D(n) \text{ \& } u' \in D'(n) \text{ \& } h(u) \neq h(u')\}$ , and let  $q'(n)$  denote the number of possible storage states of  $M$  just after the point where the input head left the second 'c' of strings in  $V'(n)$ . Then it is easily to see that  $q'(n) \leq k^{L(r(n))}$ , where  $k$  is a constant depending only on  $M$ , and  $r(n)$  is the length of each string in  $V'(n)$ . Note that  $r(n) = O(n \cdot 2^n \cdot 2^{2^n})$ . For each  $n \geq 1$ , let  $q(n)$  denote the number of possible storage states of  $M$  just after the point where the input head just left the second 'c' of strings in  $V(n)$ . Since  $M$  is a one-way machine and has only universal states, it follows that  $q(n) = q'(n) \leq k^{L(r(n))}$ . From this and the assumption that  $L(n) = o(\log n)$ , it follows that  $|V(n)| = 2^{2^n} > q(n)$  for large  $n$ . For such a large  $n$ , there must be two different strings  $x, y \in V(n)$  such that  $s(x) = s(y)$ . This contradicts Proposition 1. Thus, we complete the proof of " $L_3 \notin \text{weak-1USPACE}(o(\log n))$ ". Q.E.D.

From Theorem 1, Theorem 2, and Theorem 3, we have the following corollary.

**Corollary 1.** For any function  $\log \log n \leq L(n) = o(\log n)$ , and for each  $X \in \{\text{strong}, \text{weak}\}$  and each  $Y \in \{A, N, U, D\}$ ,  $X\text{-1YPDA}(L(n)) \not\subseteq X\text{-2YPDA}(L(n))$ .

#### 4 A relationship among determinism, nondeterminism, and alternation

This section mainly investigates a relationship among the accepting powers of one-way (or two-way) alternating pushdown automata, deterministic pushdown automata, nondeterministic pushdown automata, and alternating pushdown automata with only universal states with sublogarithmic space.

**Theorem 4.**

$$weak-1APDA(\log \log n) - (weak-NSPACE(o(\log n)) \cup weak-USPACE(o(\log n))) \neq \emptyset$$

**Proof:**

$$\text{Let } L_4 = \{Ruler_2(n)cu_1cu_2c \cdots cu_kcu \in \{0, 1, a, b, c, \#\}^+ | \\ n \geq 1 \text{ \& } k \geq 1 \text{ \& } \forall i(1 \leq i \leq k)[u_i \in D(n)] \text{ \& } \\ u \in D'(n) \text{ \& } \exists r(1 \leq r \leq k)[h(u) = h(u_r)]\}.$$

To prove the theorem, we show that

- (1)  $L_4 \in weak-1APDA(\log \log n)$ ,
  - (2)  $L_4 \notin weak-NSPACE(o(\log n))$ , and
  - (3)  $L_4 \notin weak-USPACE(o(\log n))$
- (1): The set  $L_4$  will be accepted by a weakly  $\log \log n$  space-bounded 1APDA  $M$  which acts as follows. Suppose that an input string

$$x = w_1 \# w_2 \# \cdots \# w_d cu_1 cu_2 c \cdots cu_k cu$$

is presented to  $M$ , where  $d \geq 4$ ,  $k \geq 1$ , and

- (i) for each  $s$  ( $1 \leq s \leq d$ ),  $w_s = x_{s1}t_{s1}x_{s2}t_{s2} \cdots x_{sl_s}t_{sl_s}$  (where  $l_s \geq 2$ ,  $x_{sj}'s \in \{a, b\}$ , and  $t_{sj}'s \in \{0, 1\}^+$ ),
- (ii) for each  $i$  ( $1 \leq i \leq k$ ),  $u_i = y_{i1}v_{i1}y_{i2}v_{i2} \cdots y_{il'_i}v_{il'_i}$  (where  $l'_i \geq 2$ ,  $y_{ij}'s \in \{a, b\}$ , and  $v_{ij}'s \in \{0, 1\}^+$ ),
- (iii)  $u = y_1v_1y_2v_2 \cdots y_lv_l$  (where  $l \geq 2$ ,  $y_j's \in \{a, b\}$ , and  $v_j's \in \{0, 1\}^+$ ).

(Input strings of the form different from the above can be easily rejected by  $M$ .) Let  $n = |t_{11}|$ .  $M$  makes a universal branch as follows.

- (a) In the first branch  $B_1$ ,  $M$  checks whether

$$\forall s(1 \leq s \leq d) \forall i(1 \leq i \leq k)$$

$$[|t_{s1}| = |t_{s2}| = \cdots = |t_{sl_s}| = |v_{i1}| = |v_{i2}| = \cdots = |v_{il'_i}| = |v_1| = |v_2| = \cdots = |v_l| = n].$$

This is universally done by using  $n$  space of the pushdown stack.

- (b) In the second branch  $B_2$ ,  $M$  checks whether

- (b-1) for each odd number  $s(1 \leq s \leq d)$

$$[t_{s1} = B(n, 1) = \underbrace{00 \cdots 0}_n, t_{sl_s} = B(n, 2^n) = \underbrace{11 \cdots 1}_n, \text{ and } \forall j(1 \leq j \leq l_s - 1)[\text{num}(t_{sj+1}) = \text{num}(t_{sj}) + 1],$$

where for each string  $w \in \{0, 1\}^+$ ,  $\text{num}(w)$  denotes the integer represented by the 'binary number'  $w$  with the leftmost symbol as the most significant bit.

- (b-2) for each even number  $s(1 \leq s \leq d)$

$$[t_{s1} = B(n, 1)^R = \underbrace{00 \cdots 0}_n, t_{sl_s} = B(n, 2^n)^R = \underbrace{11 \cdots 1}_n, \text{ and } \forall j(1 \leq j \leq l_s - 1)[\text{num}(t_{sj+1}^R) = \text{num}(t_{sj}^R) + 1],$$

- (b-3)  $\forall i(1 \leq i \leq k)[v_{i1} = B(n, 1), v_{il'_i} = B(n, 2^n), \text{ and } \forall j(1 \leq j \leq l'_i - 1)[\text{num}(v_{ij+1}) = \text{num}(v_{ij}) + 1]]$ , and

- (b-4)  $v_1 = B(n, 1)^R, v_l = B(n, 2^n)^R, \text{ and } \forall j(1 \leq j \leq l - 1)[\text{num}(v_{j+1}) = \text{num}(v_j) + 1]$

(b-1) is universally checked as follows.

The check of ' $t_{s1} = B(n, 1)$ ' and ' $t_{sl_s} = B(n, 2^n)$ ' is straightforward. For each odd  $s(1 \leq s \leq d)$ , in one branch,  $M$  checks that  $\text{num}(t_{sj+1}) = \text{num}(t_{sj}) + 1$  for each  $1 \leq j \leq l_s - 1$ . To do so,  $M$  makes a universal branch. For each  $j(1 \leq j \leq l_s - 1)$ , in the  $j$ -th branch,  $M$  universally checks whether  $\text{num}(t_{sj+1}) = \text{num}(t_{sj}) + 1$ . That is, for each  $m(1 \leq m \leq |t_{sj}|)$ , in the  $m$ -th branch,  $M$  stores the symbol  $t_{sj}(m)$  (where  $t_{sj}(m)$  denotes the  $m$ -th symbol (from the left) of  $t_{sj}$ ) in its finite control, stores  $Z^m$  in the pushdown stack (where  $Z$  is a pushdown stack symbol), picks up the  $m$ -th symbol  $t_{sj+1}(m)$  of  $t_{sj+1}$  by using  $Z^m$  in the pushdown stack, and enters an accepting state only if it finds out that if either  $(m = |t_{sj}|)$  or  $(m \neq |t_{sj}| \text{ \& } t_{sj}(m+1) = t_{sj}(m+2) = \cdots = t_{sj}(|t_{sj}|) = 1)$ , then  $t_{sj+1}(m) = \overline{t_{sj}(m)}$ , and otherwise,  $t_{sj+1}(m) = t_{sj}(m)$ , where  $\bar{1} = 0$  and  $\bar{0} = 1$ .

The checks of (b-2), (b-3), and (b-4) are similar to the check of (b-1).

- (c) In the third branch  $B_3$ ,  $M$  checks whether

- (c-1)  $x_{11}x_{12} \cdots x_{1l_1} = aa \cdots a$  and  $x_{d1}x_{d2} \cdots x_{dl_d} = bb \cdots b$ , and

- (c-2)  $\forall s(1 \leq s \leq d-1)[\text{num}(h(x_{s+1,1}x_{s+1,2} \cdots x_{s+1,l_{s+1}})) = \text{num}(h(x_{s1}x_{s2} \cdots x_{sl_s})) + 1]$ .

The check of (c-1) is trivially done by using the finite control. On the other hand, (c-2) is universally checked as follows.

For each  $s(1 \leq s \leq d-1)$ , in the  $s$ -th branch,  $M$  checks whether

$$\text{num}(h(x_{s+1,1}x_{s+1,2} \cdots x_{s+1,l_{s+1}})) = \text{num}(h(x_{s1}x_{s2} \cdots x_{sl_s})) + 1$$

To do so,  $M$  further makes a universal branch. For each  $j (1 \leq j \leq l_s)$ , in the  $j$ -th branch,  $M$  stores the symbol  $x_{sj}$  in the finite control, and stores the "yardstick" string  $t_{sj}$  (positioned just after  $x_{sj}$ ) in the pushdown stack. Then, by using  $t_{sj}$  stored in the pushdown stack,  $M$  tries to pick up the symbol  $x_{s+1,j}$  and check that  $x_{sj}$  and  $x_{s+1,j}$  have a desired relationship. To do so,  $M$  again makes a universal branch. That is, for each  $j' (1 \leq j' \leq l_{s+1})$ , in the  $j'$ -th branch,  $M$  stores the symbol  $x_{s+1,j'}$  in the finite control and compares  $t_{sj}$  (stored in the pushdown stack) with  $t_{s+1,j'}$ . If  $t_{s+1,j'} \neq t_{sj}^R$ , then  $M$  immediately enters an accepting state. If  $t_{s+1,j'} = t_{sj}^R$ , then  $M$  enters an accepting state only if one of the following three conditions is true.

- (i)  $j = l_s \& x_{s+1,j} = \bar{x}_{sj}$ ,
- (ii)  $j \neq l_s \& x_{sj+1} = x_{sj+2} = \dots = x_{sl_s} = b \& x_{s+1,j'} = \bar{x}_{sj}$ ,
- (iii)  $j \neq l_s \& \exists r (j+1 \leq r \leq l_s) [x_{sr} = a] \& x_{s+1,j'} = x_{sj}$

where  $\bar{a} = b$  and  $\bar{b} = a$ .

- (d) In the fourth branch  $D_4$ ,  $M$  checks whether  $h(u_r) = h(u)$ , i.e.,  $y_{r1}y_{r2} \dots y_{rl'_r} = y_1y_2 \dots y_l$  for some  $r (1 \leq r \leq k)$ .

To do so,  $M$  existentially chooses some  $r (1 \leq r \leq k)$ , moves to the segment  $u_r$ , and universally checks whether  $y_{rj} = y_j$  for each  $1 \leq j \leq l'_r$ . To check that  $y_{rj} = y_j$ ,  $M$  stores the symbol  $y_{rj}$  in the finite control, stores the "yardstick" string  $v_{rj}$  (positioned just after  $y_{rj}$ ) in the pushdown stack, existentially guesses  $j'$  (such that  $v_{j'} = v_{rj}^R$ ), picks up the symbol  $y_{j'}$ , and enters an accepting state only if it finds out that  $v_{j'} = v_{rj}^R$  and  $y_{rj} = y_{j'}$ . (Another method to check that  $y_{rj} = y_j$  is to use a technique similar to that in the last paragraph of (c) above. That is,  $M$  stores the symbol  $y_{rj}$  in the finite control, and stores the yardstick string  $v_{rj}$  (positioned just after  $y_{rj}$ ) in the pushdown stack. Then,  $M$  makes a universal branch as follows. For each  $j' (1 \leq j' \leq l)$ , in the  $j'$ -th branch,  $M$  stores the symbol  $y_{j'}$  in the finite control and compares  $v_{rj}$  (stored in the pushdown stack) with the yardstick string  $v_{j'}$ . If  $v_{j'} \neq v_{rj}^R$ , then  $M$  immediately enters an accepting state. If  $v_{j'} = v_{rj}^R$ , then  $M$  enters an accepting state only if  $y_{rj} = y_{j'}$ .)

$M$  accepts the input string  $x$  if and only if (a), (b), (c), and (d) above are all checked successfully if and only if  $x$  is in  $L_4$ . It will be obvious that each computation path in an accepting computation tree of  $M$  on  $x$  is such that the space of the pushdown stack is bounded by  $n = |t_{11}| \leq \log \log |x|$ .

- (2): Suppose that there exists a weakly  $L(n)$  space-bounded 2NTM  $M$  accepting  $L_4$ , where  $L(n) = o(\log n)$ . We assume without loss of generality that when  $M$  accepts  $x$  in  $L_4$ , it enters an accepting state on the right endmarker '\$'. For each  $n \geq 1$ , let

$$V(n) = \{ \text{Ruler}_2(n)ycu | y \in W(n) \& u \in D'(n) \}, \text{ where } W(n) = \{ cu_1cu_2c \dots cu_{2^{2^n}} | \forall i (1 \leq i \leq 2^{2^n}) [u_i \in D(n)] \}.$$

We consider the computation of  $M$  on the strings in  $V(n)$ . Let  $r(n)$  be the length of each  $x$  in  $V(n)$ . Then  $r(n) = O(n \cdot 2^n \cdot 2^{2^n})$ . Let  $s$  and  $k$  be the number of states (of the finite control) and storage-tape symbols of  $M$ . When  $M$  uses at most  $L(r(n))$  storage-cells, there will be at most  $u(n) = sL(r(n))k^{L(r(n))}$  possible storage states. We denote the set of these storage states by  $C(n) = \{q_1, q_2, \dots, q_{u(n)}\}$ . For each  $y \in W(n)$ , each  $q \in C(n)$  and each  $d \in \{r, l\}$ , let  $M_y(q, d)$  be a subset of  $(C(n) \times \{r, l\}) \cup \{H\}$  which is defined as follows ( $H$  is a new symbol):

(i)  $(q', d') \in M_y(q, d) \Leftrightarrow$  when  $M$  enters  $y$  in storage state  $q$  by moving right (if  $d = r$ ) or by moving left (if  $d = l$ ), there exists a sequence of steps of  $M$  in which  $M$  eventually exits  $y$  in storage state  $q'$  by moving left (if  $d' = l$ ) or by moving right (if  $d' = r$ )

(ii)  $H \in M_y(q, d) \Leftrightarrow$  when  $M$  enters  $y$  in storage state  $q$  by moving right (if  $d = r$ ) or by moving left (if  $d = l$ ), there exists a sequence of steps of  $M$  in which  $M$  never exits  $y$ . (Note the assumption that  $M$  never enters an accepting state in  $y$ .)

Let  $y_1, y_2$  be two strings in  $W(n)$ . We say that  $y_1$  and  $y_2$  are  $M$ -equivalent if for each  $(q, d) \in C(n) \times \{r, l\}$ ,  $M_{y_1}(q, d) = M_{y_2}(q, d)$ . Clearly,  $M$ -equivalence is an equivalence relation on strings in  $W(n)$ , and there are at most

$$E(n) = (2^{2 \cdot u(n)+1})^{2 \cdot u(n)}$$

$M$ -equivalence classes denoted by  $C_1, C_2, \dots, C_{E(n)}$ .

For each  $y = cu_1cu_2 \dots cu_{2^{2^n}}$  in  $W(n)$ , let  $b(y) = \{u \in D'(n) | \exists j (1 \leq j \leq 2^{2^n}) [h(u_j) = h(u)]\}$ .

Furthermore, for each  $n \geq 1$ , let  $R(n) = \{b(y) | y \in W(n)\}$ . Then  $|R(n)| = 2^{2^{2^n}} - 1$ . Since  $\lim_{n \rightarrow \infty} L(n)/\log n = 0$ , it follows that  $\lim_{n \rightarrow \infty} L(r(n))/\log r(n) = 0$ , and thus  $\lim_{n \rightarrow \infty} L(r(n))/2^n = 0$ . From this, it follows that  $|R(n)| > E(n)$  for large  $n$ . For such  $n$ , there must be some  $Q, Q' (Q \neq Q')$  in  $R(n)$  and some  $M$ -equivalence class  $C_i (1 \leq i \leq E(n))$  such that the following statement holds:

"There exist two strings  $y, z \in W(n)$  such that (i)  $b(y) = Q \neq Q' = b(z)$ , and (ii)  $y, z \in C_i$  (i.e.,  $y$  and  $z$  are  $M$ -equivalent.)"

Because of (i), we can, without loss of generality, assume that there is some  $u$  such that  $u \in b(y) - b(z)$ . It is clear that  $y' = \text{Ruler}_2(n)ycu$  is in  $L_4 \cap V(n)$ , so there exists an  $L(r(n))$  space-bounded accepting computation tree of  $M$  on  $y'$ . Because of (ii), from this tree, we can easily construct an  $L(r(n))$  space-bounded accepting computation tree of  $M$  on  $z' = \text{Ruler}_2(n)zcu$ . Thus, we can conclude that  $z'$  is also accepted by  $M$ . Since  $z'$  is not in  $L_4$ , we get a contradiction. This completes the proof of (2).

- (3): The proof of (3) is similar to that of (2).

Q.E.D.

From Theorem 4, we have the following corollary.

**Corollary 2.**

- (1)  $weak-1APDA(\log \log n) - (weak-2NPDA(o(\log n)) \cup weak-2UPDA(o(\log n))) \neq \emptyset$ .
- (2) For any function  $\log \log n \leq L(n) = o(\log n)$ ,
  - $weak-2NPDA(L(n)) \cup weak-2UPDA(L(n)) \not\subseteq weak-2APDA(L(n))$ ,
  - $weak-1NPDA(L(n)) \cup weak-1UPDA(L(n)) \not\subseteq weak-1APDA(L(n))$ .

**Theorem 5.**

$strong-2APDA(\log \log n) - (weak-NSPACE(o(\log n)) \cup weak-USPACE(o(\log n))) \neq \emptyset$ .

**Proof:** Let  $L_1$  be the set described in the proof of Theorem 1. By using the same technique as in the proof of Theorem 4, we can show that  $L_1 \notin weak-NSPACE(o(\log n)) \cup weak-USPACE(o(\log n))$ . On the other hand, it is shown in the proof of Theorem 1 that  $L_1$  is in  $strong-2APDA(\log \log n)$ . This completes the proof of the theorem. Q.E.D.

From Theorem 5, we have the following corollary.

**Corollary 3.**

- (1)  $strong-2APDA(\log \log n) - (weak-2NPDA(o(\log n)) \cup weak-2UPDA(o(\log n))) \neq \emptyset$ .
- (2) For any function  $\log \log n \leq L(n) = o(\log n)$ ,  $strong-2NPDA(L(n)) \cup strong-2UPDA(L(n)) \not\subseteq strong-2APDA(L(n))$ .

**Theorem 6.**

$weak-1UPDA(\log \log n) - weak-1NSPACE(o(\log n)) \neq \emptyset$ . Thus,  $weak-1UPDA(\log \log n) - weak-1NPDA(o(\log n)) \neq \emptyset$ .

**Proof.** Let  $L'_2 = \{Ruler_2(n) | n \geq 1\}$ . It is implicitly shown in the proof of Theorem 4 that  $L'_2$  is accepted by a weakly  $\log \log n$  space bounded 1UPDA. On the other hand, we can show that  $L'_2$  is not in  $weak-1NSPACE(o(\log n))$  by using the same idea as in the proof of Theorem 2. This completes the proof of Theorem 6. Q.E.D.

Statement (2) of the following corollary solves an open problem in [9].

**Corollary 4.** For any function  $\log \log n \leq L(n) = o(\log n)$ ,

- (1)  $weak-1DPDA(L(n)) \not\subseteq weak-1UPDA(L(n))$ , and
- (2)  $weak-1USPACE(L(n))$  and  $weak-1NSPACE(L(n))$  are incomparable.

**Proof.**

(1): This follows directly from Theorem 6.

(2): From Theorem 6, to prove (2), it is sufficient to show that  $weak-1NSPACE(\log \log n) - weak-1USPACE(o(\log n)) \neq \emptyset$ .

It is known that the set  $L' = \{0^n 10^m | n \neq m\}$  is in  $weak-1NSPACE(\log \log n)$  [6]. We below prove that  $L'$  is not in  $weak-1USPACE(o(\log n))$ .

Suppose that there exists a weakly  $L(n)$  space-bounded 1UTM  $M$ , which accepts  $L'$ , where  $L(n) = o(\log n)$ .

We first note that for each  $n \geq 1$ , there is at least one computation path of  $M$  on the input string  $0^n 10^n$  in which  $M$  never enters an accepting state, because of  $0^n 10^n \notin L'$ . Fix such a computation path of  $M$  on  $0^n 10^n$ , and denote it by  $p(n)$ . Let  $s(n)$  denote the storage state of  $M$  just after the point where in  $p(n)$  the input head left the symbol '1' of  $0^n 10^n$ . Then, the following proposition must hold.

**Proposition 2.** For any two different strings  $0^n 10^n$  and  $0^m 10^m$  ( $n \neq m$ ),  $s(n) \neq s(m)$ .

[**Proof.** For otherwise, suppose that  $s(n) = s(m)$ . Then, there would be a computation path of  $M$  on the string  $0^n 10^m$  in which  $M$  never enters an accepting state. This means that  $0^n 10^m$  is rejected by  $M$ . This contradicts the fact that  $0^n 10^m$  is in  $L'$ .]

**Proof of Corollary 4 (continued).**

For each  $n \geq 2$ , let  $V'(n) = \{0^i 10^j | 2 \leq i \leq n\}$ , and let  $q'(n)$  denote the number of possible storage states of  $M$  just after the point where the input head left the symbol '1' of strings in  $V'(n)$ . Then it is easy to see that for infinitely many  $n$ ,  $q'(n) \leq r^{L(n+2)}$  (where  $r$  is a constant depending only on  $M$ ). For each  $n \geq 2$ , let  $V(n) = \{0^i 10^i | 2 \leq i \leq n\}$ , and let  $q(n)$  denote the number of possible storage states of  $M$  just after the point where the input head left the symbol '1' of strings in  $V(n)$ . Noting that  $M$  is a one-way machine and has only universal states, we can easily see that  $q(n) = q'(n)$ . Since  $L(n) = o(\log n)$ , it follows that  $n > q(n)$  for some large  $n$ . For such a large  $n$ , there must be two integers  $n_1, n_2$  such that (i)  $2 \leq n_1 < n_2 \leq n$  and (ii)  $s(n_1) = s(n_2)$ . This contradicts Proposition 2. Thus, we complete the proof of " $L' \notin weak-1USPACE(o(\log n))$ ". Q.E.D.

Unfortunately, it is unknown whether  $weak-1UPDA(L(n))$  and  $weak-1NPDA(L(n))$  are incomparable for any  $\log \log n \leq L(n) = o(\log n)$ .

## 5 Closure Properties

This section shows that for any function  $\log \log n \leq L(n) = o(\log n)$ , (1)  $weak-1APDA(L(n))$ ,  $weak-1UPDA(L(n))$ , and  $X-YPDA(L(n))$  ( $X \in \{strong, weak\}$ , and  $Y \in \{2N, 2U, 2D\}$ ) are not closed under concatenation, Kleene closure, and length preserving homomorphism, and (2)  $weak-1UPDA(L(n))$  is not closed under complementation.

**Lemma 1.** Let



- $L_5 = \{Ruler_2(n)cucuc_1cu_2c \dots cu_k \in \{0, 1, a, b, c, \#\}^+ \mid$   
 $n \geq 1 \ \& \ k \geq 1 \ \& \ \forall i(1 \leq i \leq k)[u_i \in D(n)]$   
 $\& \ u \in D'(n) \ \& \ h(u) = h(u_k)\}$ ,
- $L_6 = \{cy_1z_1y_2z_2 \dots y_kz_k \in \{0, 1, a, b, c\}^+ \mid k \geq 2 \ \&$   
 $\forall j(1 \leq j \leq k)[y_j \in \{a, b\} \ \& \ z_j \in \{0, 1\}^*]\}^*$ ,
- $L_7 = L_5 \cup L_6$ ,
- $L_8 = \{Ruler_2(n)cu_1cu_2c \dots cu_k \in \{0, 1, a, b, c, \#\}^+ \mid$   
 $n \geq 1 \ \& \ k \geq 2 \ \& \ \forall i(1 \leq i \leq k)[u_i \in D(n)]\}$ , and
- $L_9 = \{Ruler_2(n)cuc_1u_1c_2u_2 \dots c_ku_k \in \{0, 1, a, b, c, d, \#\}^+ \mid$   
 $n \geq 1 \ \& \ k \geq 1 \ \& \ u \in D'(n) \ \& \ \forall i(1 \leq i \leq k)[u_i \in D(n) \ \& \ c_i \in \{c, d\}] \ \&$   
 $\exists j(1 \leq j \leq k)[c_j = d \ \& \ \forall i(1 \leq i \leq k, i \neq j)[c_i = c] \ \& \ h(u) = h(u_j)]\}$ .

Then,  $L_5, L_6, L_7, L_8, L_9$  are all in  $weak-1UPDA(\log \log n)$ , and in  $strong-2DPDA(\log \log n)$ .

**Proof.** By using a technique similar to that in the proof of Theorem 4, we can easily show that each  $L_i (5 \leq i \leq 9)$  is accepted by a weakly  $\log \log n$  space-bounded 1UPDA, and thus  $L_i \in weak-1UPDA(\log \log n)$ .

The proof of " $L_i \in strong-2DPDA(\log \log n)$  for each  $i \in \{5, 6, 7, 8, 9\}$ " is left to the reader. Q.E.D.

**Lemma 2.** Let

$$L_{10} = \{Ruler_2(n)cucuc_1cu_2c \dots cu_k \in \{0, 1, a, b, c, \#\}^+ \mid n \geq 1 \ \& \ k \geq 1 \ \& \ u \in D'(n) \ \& \ \forall i(1 \leq i \leq k)[u_i \in D(n)] \ \& \ \exists r(1 \leq r \leq k)[h(u) = h(u_r)]\}.$$

Then  $L_{10} \notin weak-1APDA(o(\log n)) \cup weak-2NPDA(o(\log n)) \cup weak-2UPDA(o(\log n))$ .

**Proof.** By using the same technique as in the proof of Theorem 1 (Theorem 4), We can show that  $L_{10} \notin weak-1APDA(o(\log n)) (L_{10} \notin weak-2NPDA(o(\log n)) \cup weak-2UPDA(o(\log n)))$ . Q.E.D.

**Theorem 7.** For any function  $\log \log n \leq L(n) = o(\log n)$ ,  $weak-1APDA(L(n))$ ,  $weak-1UPDA(L(n))$ , and  $X-YPDA(L(n))$  (where  $X \in \{strong, weak\}$  and  $Y \in \{2N, 2U, 2D\}$ ) are not closed under concatenation, Kleene closure, and length preserving homomorphism.

**Proof.** Let  $L_5, L_6, L_7, L_8, L_9, L_{10}$  be the set described above. We first observe that

- (i)  $L_5L_6 \cap L_8 = L_{10}$ , and
- (ii)  $weak-1APDA(L(n))$ ,  $weak-1UPDA(L(n))$ , and  $X-YPDA(L(n))$  (where  $X \in \{strong, weak\}$  and  $Y \in \{2N, 2U, 2D\}$ ) are closed under intersection.

Nonclosure under concatenation follows from this observation and Lemma 1 and 2. Nonclosure under Kleene closure follows from (ii) above, Lemma 1, Lemma 2, and the fact that  $L_7^* \cap L_8 = L_{10}$ .

Let  $g$  be a length preserving homomorphism such that  $g(0) = 0$ ,  $g(1) = 1$ ,  $g(c) = c$ ,  $g(\#) = \#$ , and  $g(d) = c$ . Then,  $g(L_9) = L_{10}$ . From this and from Lemma 1 and 2, nonclosure under length preserving homomorphism follows. Q.E.D.

**Theorem 8.** For any function  $\log \log n \leq L(n) = o(\log n)$ ,  $weak-1UPDA(L(n))$  is not closed under complementation.

**Proof.** Let  $L_{11} = \{Ruler_2(n)cucuc' \mid n \geq 1 \ \& \ u \in D(n) \ \& \ u' \in D'(n) \ \& \ h(u) = h(u')\}$ . By using a technique similar to that in the proof of Theorem 4, we can show that  $L_{11}$  is accepted by a weakly  $\log \log n$  space-bounded 1UPDA. On the other hand, by using a technique similar to that in the proof of Theorem 3, we can show that  $\overline{L_{11}}$  (the complement of  $L_{11}$ ) is not in  $weak-1USPACE(o(\log n))$ , and thus  $\overline{L_{11}}$  is not in  $weak-1UPDA(o(\log n))$ . This completes the proof of the theorem. Q.E.D.

## 6 Weak versus Strong

This section briefly discusses a relationship between 'strongly' and 'weakly'.

It is shown in [4] that  $strong-1ASPACE(o(\log n))$  is equal to the class of regular languages. Let  $L_5$  be the set described in Lemma 1. Clearly,  $L_5$  is not a regular language. On the other hand,  $L_5 \in weak-1UPDA(\log \log n)$ . From this observation, we have the following theorem.

**Theorem 9.** For any function  $\log \log n \leq L(n) = o(\log n)$ ,

- (1)  $strong-1UPDA(L(n)) \not\subseteq weak-1UPDA(L(n))$ , and
- (2)  $strong-1APDA(L(n)) \not\subseteq weak-1APDA(L(n))$ .

It is known that the set  $L' = \{0^n 10^m \mid n \neq m\}$  is in  $weak-DSPACE(\log \log n)$  [1], and  $L'$  is in  $weak-1NSPACE(\log \log n)$  [6], but  $L'$  is not in  $strong-ASPACE(o(\log n))$  [15]. Thus, we have, for any function  $\log \log n \leq L(n) = o(\log n)$ ,

- $strong-XSPACE(L(n)) \not\subseteq weak-XSPACE(L(n))$  ( $X \in \{A, N, U, D\}$ ), and
- $strong-1NSPACE(L(n)) \not\subseteq weak-1NSPACE(L(n))$ .

Unfortunately, it is unknown whether a similar result holds also for pushdown automata.

## 7 Conclusions

We conclude this paper by giving several open problems. Below,  $L(n)$  denotes any function such that  $\log \log n \leq L(n) = o(\log n)$ .

- (1)
  - Is  $\text{weak-1APDA}(L(n))$  incomparable with  $\text{weak-2DPDA}(L(n))$ ,  $\text{weak-2NPDA}(L(n))$ , and  $\text{weak-2UPDA}(L(n))$ ?
  - Is  $\text{weak-1UPDA}(L(n))$  incomparable with  $\text{weak-2DPDA}(L(n))$ , and  $\text{weak-2NPDA}(L(n))$ ?
  - Is  $\text{weak-1NPDA}(L(n))$  incomparable with  $\text{weak-2DPDA}(L(n))$ , and  $\text{weak-2UPDA}(L(n))$ ?
- (2)
  - $\text{weak-2DPDA}(L(n)) \not\subseteq \text{weak-2UPDA}(L(n))$ ?
  - $\text{weak-2DPDA}(L(n)) \not\subseteq \text{weak-2NPDA}(L(n))$ ?
  - Is  $\text{weak-2UPDA}(L(n))$  incomparable with  $\text{weak-2NPDA}(L(n))$ ?
- (3)
  - $\text{strong-2DPDA}(L(n)) \not\subseteq \text{strong-2UPDA}(L(n))$ ?
  - $\text{strong-2DPDA}(L(n)) \not\subseteq \text{strong-2NPDA}(L(n))$ ?
  - Is  $\text{strong-2UPDA}(L(n))$  incomparable with  $\text{strong-2NPDA}(L(n))$ ?
- (4) for each  $X \in \{D, N, U, A\}$ ,  $\text{strong-2XPDA}(L(n)) \not\subseteq \text{weak-2XPDA}(L(n))$ ?
- (5) Are  $\text{weak-2APDA}(L(n))$ ,  $\text{weak-2UPDA}(L(n))$ ,  $\text{weak-2NPDA}(L(n))$ , and  $\text{weak-1APDA}(L(n))$  closed under complementation? (It is shown in [2] that  $\text{weak-USPACE}(L(n))$  and  $\text{weak-NSPACE}(L(n))$  are not closed under complementation.)
- (6) Are  $\text{strong-2APDA}(L(n))$ ,  $\text{strong-2UPDA}(L(n))$ ,  $\text{strong-2NPDA}(L(n))$ , and  $\text{strong-2DPDA}(L(n))$  closed under complementation? (It is shown in [2] that  $\text{strong-DSPACE}(L(n))$  is closed under complementation. Whether  $\text{strong-ASPACE}(L(n))$ ,  $\text{strong-USPACE}(L(n))$ , and  $\text{strong-NSPACE}(L(n))$  are closed under complementation is still an open problem.)

## References

- [1] H. Alt and K. Melhorn, "Lower bounds for the space complexity of context-free recognition," in Proc. 3rd ICALP, 1976, pp.339-354.
- [2] B. V. Braunnmühl, R. Gengler, and R. Rettinger, "The alternation hierarchy for sublogarithmic space is infinite", in Comput. Complexity, vol.3, pp.207-230,1993.
- [3] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer, "Alternation", in J. ACM, vol. 28, no.1, pp.114-133, 1981.
- [4] J. H. Chang, O. H. Ibarra, and Ravikumar, "Some observations concerning Turing machines using small space", in Information Processing Letters, vol. 25, PP. 1-9, 1987 (Erratum: Information Processing Letters, vol. 27, P. 53, 1988.)
- [5] P. Duris and Z. Galil, "On reversal-bounded counter machines and on pushdown automata with a bound on the size of the pushdown store", in Information and Control, vol. 54, pp. 217-227, 1982.
- [6] R. Freivalds, "On time complexity of deterministic and nondeterministic Turing machines", in Latvian Math. vol 23, pp.158-165, 1979.
- [7] J. Gabarro, "Pushdown space complexity and related full-A.F.L.s", in Lecture notes in Computer Science 166 (Springer-Verlag), pp. 250-259, 1984.
- [8] N. Immerman, "NSPACE is closed under complement", in SIAM J. on Computing, vol. 17, pp. 935-938, 1988.
- [9] A. Ito, K. Inoue, and I. Takanami, "A note on alternating Turing machines using small space", in Trans. IEICE, vol. E-70, no. 10, pp. 990-996, 1987.
- [10] K. Iwama. "ASPACE( $o(\log \log n)$ ) is regular", in SIAM J. on Computing, vol. 22, pp.136-146,1993.
- [11] K. N. King, "Alternating multihead finite automata", in Theoretical Computer Science, vol. 61, pp.149-174, 1988.
- [12] R. E. Lader, R. J. Lipton and L. J. Stockmeyer, "Alternating pushdown automata", in SIAM J. on Computing, vol. 13, pp.135-155,1984.
- [13] M. Liśkiewicz, and R. Reischuk, "The complexity world below logarithmic space", in Proceedings of the Ninth Annual Structure in Complexity Theory Conference, pp. 64-78, 1994.
- [14] R. Szelepcsényi, "The method of forced enumeration for nondeterministic automata", in Acta Informatica, vol. 26, pp. 279-284, 1988.
- [15] A. Szepietowski, "Alternating weak mode of space complexity is more powerful than the strong one", unpublished manuscript, 1992.
- [16] T. Yoshinaga, and K. Inoue, "A note on alternating multi-counter automata with small space", Technical Report of IEICE, COMP 94-36 1994-09.